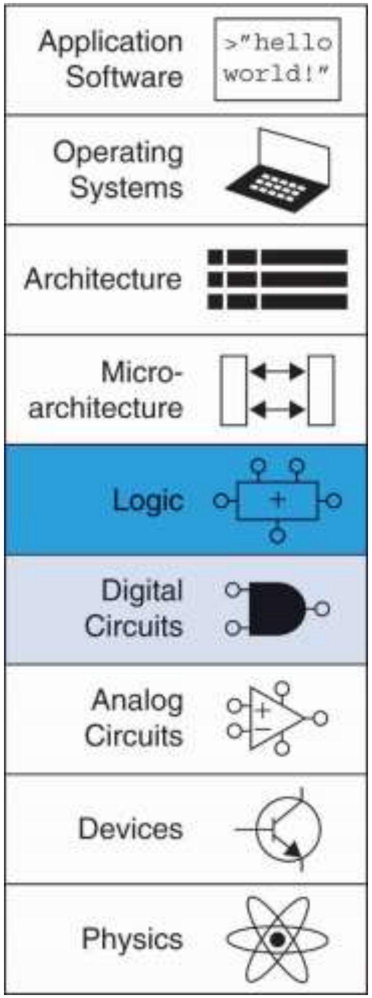
Digital Design & Computer Architecture

Sarah Harris & David Harris

Chapter 2: Combinational Logic Design

Chapter 2 :: Topics

- **Combinational Circuits**
- **Boolean Equations**
- **Boolean Algebra**
- **From Logic to Gates**
- **X's and Z's, Oh My**
- **Karnaugh Maps**
- **Combinational Building Blocks**
- **Timing**

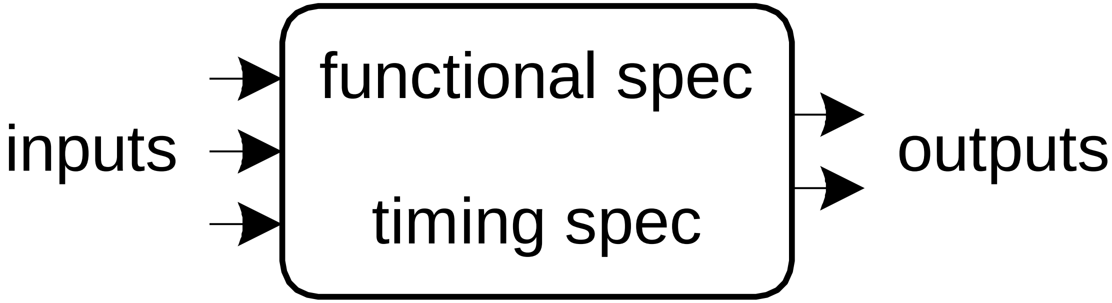


Combinational Circuits

Introduction

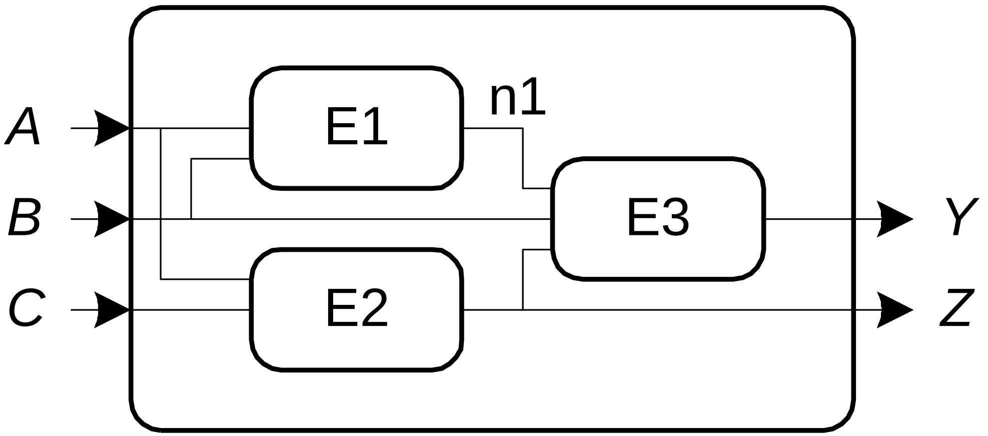
A logic circuit is composed of:

- Inputs
- Outputs
- Functional specification
- Timing specification



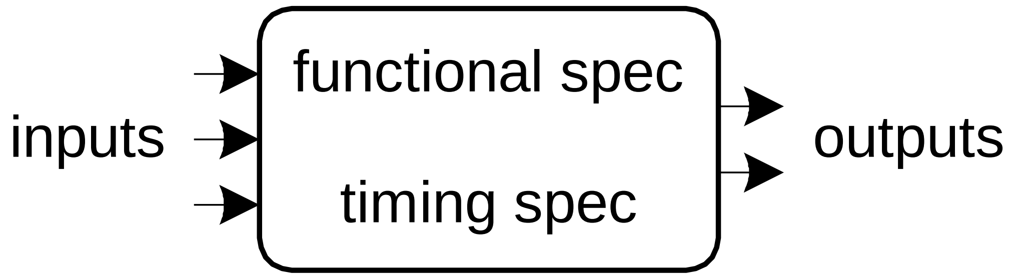
Circuits

- **Nodes**
  - Inputs: *A*, *B*, *C*
  - Outputs: *Y*, *Z*
  - Internal: n1
- **Circuit elements**
  - E1, E2, E3
  - Each a circuit



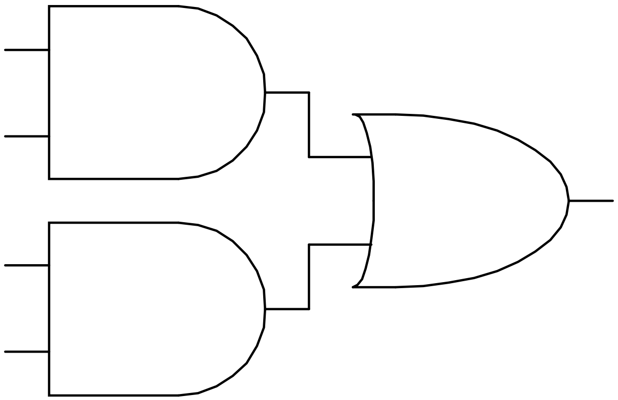
Types of Logic Circuits

- **Combinational Logic**
  - Memory*less*
  - Outputs determined by current values of inputs
- **Sequential Logic**
  - Has memory
  - Outputs determined by previous and current values of inputs



Rules of Combinational Composition

- Every element is combinational
- Every node is either an input or connects to *exactly one* output
- The circuit contains no cyclic paths
- **Example:**

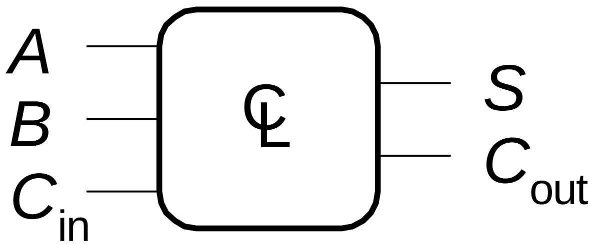


Boolean Equations

Boolean Equations

- Functional specification of outputs in terms of inputs
- **Example:** $S = F(A, B, C_{in})$

  $C_{out} = F(A, B, C_{in})$



$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

Some Definitions

- **Complement**: variable with a bar over it
  $\bar{A}, \bar{B}, \bar{C}$
- **Literal**: variable or its complement
  $A, \bar{A}, B, \bar{B}, C, \bar{C}$
- **Implicant**: product of literals
  $AB\bar{C}, \bar{A}C, BC$
- **Minterm**: product that includes all input variables
  $AB\bar{C}, A\bar{B}\bar{C}, ABC$
- **Maxterm**: sum that includes all input variables
  $(A+\bar{B}+C), (\bar{A}+B+\bar{C}), (\bar{A}+\bar{B}+C)$

Sum-of-Products (SOP) Form

- All Boolean equations can be written in SOP form
- Each row has a **minterm**
- A minterm is a **product** (AND) of literals
- Each minterm is **TRUE** for that row (and only that row)
- Form function by **ORing minterms** where output is **1**
- Thus, a **sum** (OR) of **products** (AND terms)

A	*B*	*Y*	minterm	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

$$Y = \mathrm{F}(A, B) =$$

Sum-of-Products (SOP) Form

- All Boolean equations can be written in SOP form
- Each row has a **minterm**
- A minterm is a **product** (AND) of literals
- Each minterm is **TRUE** for that row (and only that row)
- Form function by **ORing minterms** where output is **1**
- Thus, a **sum** (OR) of **products** (AND terms)

A	*B*	*Y*	minterm	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

$$Y = F(A, B) = \overline{A}B + AB = \Sigma(1, 3)$$

Long-hand Short-hand

Product-of-Sums (POS) Form

- All Boolean equations can be written in POS form
- Each row has a **maxterm**
- A maxterm is a **sum** (OR) of literals
- Each maxterm is **FALSE** for that row (and only that row)
- Form function by **ANDing maxterms** where output is **0**
- Thus, a **product** (AND) of **sums** (OR terms)

A	B	Y	maxterm	maxterm name
0	0	**0**	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	**0**	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

$$Y = \mathrm{F}(A, B) = (A + B)\bullet(\overline{A} + B) = \Pi(0, 2)$$

Long-hand Short-hand

Boolean Equations Example

- You are going to the cafeteria for lunch
  - You won't eat lunch (E = 0)
    - If it's not clean (C = 0) or
    - If they only serve meatloaf (M = 1)
- Write a truth table for determining if you will eat lunch (E).

C	M	E
0	0	
0	1	
1	0	
1	1	

SOP & POS Form

SOP – sum-of-products

C	M	E	minterm
0	0	0	$\overline{C}\ \overline{M}$
0	1	0	$\overline{C}\ M$
1	0	1	$C\ \overline{M}$
1	1	0	$C\ M$

POS – product-of-sums

C	M	E	maxterm
0	0	0	$C + M$
0	1	0	$C + \overline{M}$
1	0	1	$\overline{C} + M$
1	1	0	$\overline{C} + \overline{M}$

SOP & POS Form

SOP – sum-of-products

C	M	E	minterm
0	0	0	$\overline{C}\ \overline{M}$
0	1	0	$\overline{C}\ M$
1	0	1	$C\ \overline{M}$
1	1	0	$C\ M$

$$E = C\overline{M}$$
$$= \Sigma(2)$$

POS – product-of-sums

C	M	E	maxterm
0	0	0	$C + M$
0	1	0	$C + \overline{M}$
1	0	1	$\overline{C} + M$
1	1	0	$\overline{C} + \overline{M}$

$$E = (C + M)(C + \overline{M})(\overline{C} + \overline{M})$$
$$= \Pi(0, 1, 3)$$

Forming Boolean Expressions

Example 1:

We will go to the Park (***P*** is the output) if it's not Raining ($\overline{\boldsymbol{R}}$) and we have Sandwiches (***S***).

Boolean Equation:

Forming Boolean Expressions

Example 2:

You will be considered a Winner (**W** is the output) if we send you a Million dollars (**M**) or a small Notepad (**N**).

Boolean Equation:

Forming Boolean Expressions

Example 3:

You can Eat delicious food (***E*** is the output) if you Make it yourself (***M***) or you have a personal Chef (***C***) and she/he is talented (***T***) but not eXpensive ($\overline{X}$).

Boolean Equation:

Forming Boolean Expressions

Example 4:

You can Enter the building if you have a Hat and Shoes on or if you have a Hat on.

Boolean Equation:

Forming Boolean Expressions

Example 5:

You can Enter the building if you have a Hat and Shoes on or if you have a Hat and no Shoes on.

Boolean Equation:

Boolean Algebra: Axioms

Boolean Algebra

- Axioms and theorems to **simplify** Boolean equations
- Like regular algebra, but simpler: variables have only two values (1 or 0)
- **Duality** in axioms and theorems:
  - ANDs and ORs, 0's and 1's interchanged

Boolean Axioms

Number	Axiom	Name
A1	$B = 0$ if $B \neq 1$	Binary Field
A2	$\overline{0} = 1$	NOT
A3	$0 \bullet 0 = 0$	AND/OR
A4	$1 \bullet 1 = 1$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	AND/OR

Boolean Axioms

Number	Axiom	Dual	Name
A1	B = 0 if B ≠ 1	B = 1 if B ≠ 0	Binary Field
A2	$\overline{0} = 1$	$\overline{1} = 0$	NOT
A3	0 • 0 = 0	1 + 1 = 1	AND/OR
A4	1 • 1 = 1	0 + 0 = 0	AND/OR
A5	0 • 1 = 1 • 0 = 0	1 + 0 = 0 + 1 = 1	AND/OR

Dual: Replace: • with +
0 with 1

Boolean Algebra: Theorems of One Variable

Boolean Theorems of One Variable

Number	Theorem	Name
T1	$B \bullet 1 = B$	Identity
T2	$B \bullet 0 = 0$	Null Element
T3	$B \bullet B = B$	Idempotency
T4	$\overline{\overline{B}} = B$	Involution
T5	$B \bullet \overline{B} = 0$	Complements

Dual: Replace: $\bullet$ with +
0 with 1

Boolean Theorems of One Variable

Number	Theorem	Dual	Name
T1	$B \bullet 1 = B$	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	$B + B = B$	Idempotency
T4	$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	$B + \overline{B} = 1$	Complements

Dual: Replace: $\bullet$ with $+$
0 with 1

T1: Identity Theorem

- $B \cdot 1 = B$
- $B + 0 = B$

T2: Null Element Theorem

- $B \bullet 0 = 0$
- $B + 1 = 1$

T3: Idempotency Theorem

- $B \bullet B = B$
- $B + B = B$

T4: Involution Theorem

- $\overline{\overline{B}} = B$

T5: Complement Theorem

- $B \bullet \overline{B} = 0$
- $B + \overline{B} = 1$

Recap: Basic Boolean Theorems

Number	Theorem	Dual	Name
T1	$B \bullet 1 = B$	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	$B + B = B$	Idempotency
T4	$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	$B + \overline{B} = 1$	Complements

Boolean Algebra: Theorems of Several Variables

Boolean Theorems of Several Vars

#	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Commutativity
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	$\boxed{\text{B + (C•D) = (B+C) (B+D)}}$	Distributivity
T9	B • (B+C) = B	B + (B•C) = B	Covering
T10	(B•C) + (B•$\overline{\text{C}}$) = B	(B+C) • (B+$\overline{\text{C}}$) = B	Combining
T11	(B•C) + ($\overline{\text{B}}$•D) + (C•D) = (B•C) + ($\overline{\text{B}}$•D)	(B+C) • ($\overline{\text{B}}$+D) • (C+D) = (B+C) • ($\overline{\text{B}}$+D)	Consensus

Warning: T8' differs from traditional algebra:
OR (+) distributes over AND (•)

How to Prove

- **Method 1:** Perfect induction
- **Method 2:** Use other theorems and axioms to simplify the equation
  - Make one side of the equation look like the other

Proof by Perfect Induction

- Also called: **proof by exhaustion**
- Check every possible input value
- If the two expressions produce the same value for every possible input combination, the expressions are equal

T9: Covering

Number	Theorem	Name
T9	$B \bullet (B+C) = B$	Covering

Prove true by:

- **Method 1:** Perfect induction
- **Method 2:** Using other theorems and axioms

T9: Covering

Number	Theorem	Name
T9	B• (B+C) = B	Covering

Method 1: Perfect Induction

B	*C*	*(B+C)*	*B(B+C)*
0	0		
0	1		
1	0		
1	1		

T9: Covering

Number	Theorem	Name
T9	$B \bullet (B+C) = B$	Covering

Method 2: Prove true using other axioms and theorems.

T10: Combining

Number	Theorem	Name
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	Combining

Prove true using other axioms and theorems:

De Morgan's Theorem: Dual

#	Theorem	Dual	Name
T12	$\overline{B \bullet C \bullet D\ldots} = \overline{B} + \overline{C} + \overline{D}\ldots$	$\overline{B + C + D\ldots} = \overline{B} \bullet \overline{C} \bullet \overline{D}\ldots$	De Morgan's Theorem

The **complement** of the **product** is the **sum** of the **complements**.

Recap: Theorems of Several Vars

#	Theorem	Dual	Name
T6	$B \bullet C = C \bullet B$	$B+C = C+B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	$(B + C) + D = B + (C + D)$	Associativity
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	$B + (C \bullet D) = (B+C)\,(B+D)$	Distributivity
T9	$B \bullet (B+C) = B$	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	$(B+C) \bullet (B+\overline{C}) = B$	Combining
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) =$ $(B \bullet C) + (\overline{B} \bullet D)$	$(B+C) \bullet (\overline{B}+D) \bullet (C+D) =$ $(B+C) \bullet (\overline{B}+D)$	Consensus
T12	$\overline{B \bullet C \bullet D \ldots} = \overline{B}+\overline{C}+\overline{D}\ldots$	$\overline{B+C+D\ldots} = \overline{B} \bullet \overline{C} \bullet \overline{D}\ldots$	De Morgan’s

Boolean Algebra: Simplifying Equations

Simplifying an Equation

Simplifying may mean minimal sum of products form:

- SOP form that has the **fewest number of implicants**, where each implicant has the **fewest literals**
  - **Implicant:** product of literals

    $A\overline{B}C$, $A\overline{C}$, $B\overline{C}$
  - **Literal:** variable or its complement

    A, $\overline{A}$, B, $\overline{B}$, C, $\overline{C}$

Simplifying could also mean fewest number of gates, lowest cost, lowest power, etc. For example, Y = A xor B is likely simpler than minimal Sum of Products Y = AB + AB. These depend on details of the technology.

Simplifying Boolean Equations

Example 1:

$$Y = \bar{A}B + AB$$

$Y = B$ T10: Combining

or

$Y = B(A + \bar{A})$ T8: Distributivity

$= B(1)$ T5': Complements

$= B$ T1: Identity

Simplifying Boolean Equations

Example 2:

$$Y = A\overline{B}C + ABC + \overline{A}BC$$

$$= A\overline{B}C + ABC + ABC + \overline{A}BC \qquad \text{T3': Idempotency}$$

$$= (A\overline{B}C + ABC) + (ABC + \overline{A}BC) \qquad \text{T7': Associativity}$$

$$= AC + BC \qquad \text{T10: Combining}$$

Extra Examples

Boolean Algebra: Simplifying Equations

Simplification methods

- **Distributivity (T8, T8')** $B(C+D) = BC + BD$
  $B + CD = (B+C)(B+D)$
- **Covering (T9')** $A + AP = A$
- **Combining (T10)** $P\overline{A} + PA = P$
- **Expansion** $P = P\overline{A} + PA$
  $A = A + AP$
- **Idempotency (duplication)** $A = A + A$
- **"Simplification" theorem** $A + \overline{A}P = A + P$
  $\overline{A} + AP = \overline{A} + P$

Proving the "Simplification" Theorem

"Simplification" theorem

$$A + \overline{A}P = A + P$$

Method 1:

Method 2:

T11: Consensus

Number	Theorem	Name
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) = (B \bullet C) + (\overline{B} \bullet D)$	Consensus

Prove using other theorems and axioms:

Simplification methods

- **Distributivity (T8, T8')** $B(C+D) = BC + BD$

  $B + CD = (B+C)(B+D)$
- **Covering (T9')** $A + AP = A$
- **Combining (T10)** $\mathbf{P\overline{A} + PA = P}$
- **Expansion** $P = P\overline{A} + PA$

  $A = A + AP$
- **Idempotency (duplication)** $A = A + A$
- **"Simplification" theorem** $A + \overline{A}P = A + P$

  $\overline{A} + AP = \overline{A} + P$

Simplification methods

- **Distributivity (T8, T8')** $B(C+D) = BC + BD$
  $B + CD = (B+C)(B+D)$
- **Covering (T9')** $A + AP = A$
- Combining (T10) $P\overline{A} + PA = P$
- **Expansion** $P = P\overline{A} + PA$
  $A = A + AP$
- **Idempotency (duplication)** $A = A + A$
- **"Simplification" theorem** $A + \overline{A}P = A + P$
  $\overline{A} + AP = \overline{A} + P$

Simplifying Boolean Equations

Example 3:

$$Y = A(AB + ABC)$$

Simplification methods

- **Distributivity (T8, T8')** $B(C+D) = BC + BD$

  $B + CD = (B+C)(B+D)$

- **Covering (T9')** $A + AP = A$

- **Combining (T10)** $P\overline{A} + PA = P$

- **Expansion** $P = P\overline{A} + PA$

  $A = A + AP$

- **Idempotency (duplication)** $A = A + A$

- **"Simplification" theorem** $A + \overline{A}P = A + P$

  $\overline{A} + AP = \overline{A} + P$

Simplifying Boolean Equations

Example 4:

$Y = A'BC + A'$

Recall: $A' = \bar{A}$

Simplifying Boolean Equations

Example 4:

$Y = A'BC + A'$

Recall: $A' = \bar{A}$

or

Multiplying Out: SOP Form

An expression is in **sum-of-products (SOP)** form when all products contain literals only.

- **SOP form:** Y = AB + BC' + DE
- **NOT SOP form:** Y = DF + E(A'+B)
- **SOP form:** Z = A + BC + DE'F

Multiplying Out: SOP Form

Example 5:

$$Y = (A + C + D + E)(A + B)$$

Apply T8' first when possible: W+XZ = (W+X)(W+Z)

or

Simplifying Boolean Equations

Example 6:

$Y = AB + BC + B'D' + AC'D'$

Method 1:

Method 2:

Literal and implicant ordering

- Variables within an implicant should be in alphabetical order.
- The order of implicants doesn’t matter.

Simplifying Boolean Equations

Example 7:

$Y = (A + BC)(A + DE)$

Apply T8' first when possible: $W+XZ = (W+X)(W+Z)$

or

Review: Canonical SOP & POS Forms

SOP – sum-of-products

$E = \boxed{C\overline{M}}$

C	M	E	minterm
0	0	0	$\overline{C}\ \overline{M}$
0	1	0	$\overline{C}\ M$
1	0	1	$C\ \overline{M}$
1	1	0	$C\ M$

same

POS – product-of-sums

$E = (C + M)(C + \overline{M})(\overline{C} + \overline{M})$

C	M	E	maxterm
0	0	0	$C + M$
0	1	0	$C + \overline{M}$
1	0	1	$\overline{C} + M$
1	1	0	$\overline{C} + \overline{M}$

Factoring: POS Form

An expression is in **product-of-sums (POS)** form when all sums contain literals only.

- **POS form:** $Y = (A+B)(C+D)(E'+F)$
- **NOT POS form:** $Y = (D+E)(F'+GH)$
- **POS form:** $Z = A(B+C)(D+E')$

Factoring: POS Form

Example 8:

$Y = (A + B'CDE)$

Apply T8' first when possible: $W+XZ = (W+X)(W+Z)$

Factoring: POS Form

Example 9:

$Y = AB + C'DE + F$

Apply T8' first when possible: $W+XZ = (W+X)(W+Z)$

De Morgan's Theorem

Example 10:

$$Y = \overline{(A+\overline{BD})\overline{C}}$$

- Work from the **outside in** (i.e., top bar, then down)
- Use **involution** when possible

De Morgan's Theorem

Example 11:

$$Y = \overline{(\overline{ACE}+\overline{D}) + B}$$

Common Errors

Boolean Algebra: Simplifying Equations

Common Errors

- Using ticks ‘ instead of **bars** over variables when writing equations by hand – ticks are easy to lose
- **Not** keeping terms **aligned** from step to step
  - Alignment helps you see what changed from step-to-step.
  - It helps in both solving and double-checking the problem.
- Applying **multiple theorems** to the same term in one step
- Applying **your own personal theorems** – don’t do it ☺
- And, on a related note: ***almost*** applying the correct theorem
- ***Not*** looking for opportunities to use combining, covering, and distributivity (especially the dual form).

Common Errors

- **Losing bars** (alignment will help you avoid this)
- **Losing terms** (alignment will help you avoid this)
- **Trying to do multiple steps at once** – this is prone to errors!
- **Applying theorems incorrectly**, for example:
  - **Wrong:** $ABC + \bar{A}B\bar{C} = B$ **Correct:** $ABC + A\bar{B}C = AC$. Products may only **differ in a single term** when using the combining theorem.
  - **Wrong:** $(A + \bar{A}) = 0$ **Correct:** $A + \bar{A} = 1$
  - **Wrong:** $(A \bullet \bar{A}) = 1$ **Correct:** $A \bullet \bar{A} = 0$
  - **Wrong:** ABC = B **Correct:** **B** + ABC = B. In order to use the covering theorem, you must have a term that covers the other terms.
  - **Wrong:** $\overline{AC} = \bar{A}\bar{C}$ **Correct:** $\overline{AC} = \bar{A} + \bar{C}$ (De Morgan’s)
  - **Wrong:** $\overline{A + C} = \bar{A} + \bar{C}$ **Correct:** $\overline{A + C} = \bar{A}\bar{C}$ (De Morgan’s)

Common Errors with De Morgan’s

- Not starting from the outside parentheses and working in: this often causes additional steps.
- Trying to apply De Morgan’s theorem to an entire **complex operation** (instead of just to terms ANDed under a bar or terms ORed under a bar)
- **Losing bars.** Remember that applying the De Morgan’s Theorem is a 3 step process. For a product term under a bar:
  1. Change ANDs to ORs (or vice versa for a sum term under a bar)
  2. Bring down the terms
  3. Put bars over the individual terms
- Not keeping terms associated (i.e., **losing parentheses**)
  - For example, $\overline{ABC} = (\overline{A}+\overline{B}+\overline{C})$
  - Example error:
    - **Wrong:** (ABC)’C+D’ = **A’+B’+C’**C + D’ = **A’ + B’** + D’
    - **Correct:** (ABC)’C + D’ = **(A’+B’+C’)**C + D’ = **A’C+B’C** + D’

From Logic to Gates

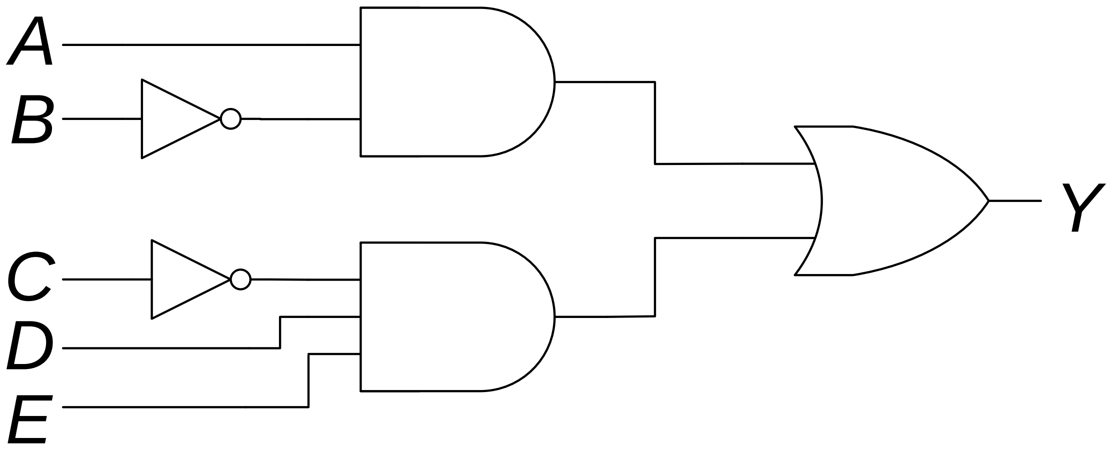
From Logic to Gates

Build the following equation using logic gates:

$$Y = A\overline{B} + \overline{C}DE$$

Circuit Schematics Rules

- Inputs on the left (or top)
- Outputs on right (or bottom)
- Gates flow from left to right
- Straight wires are best



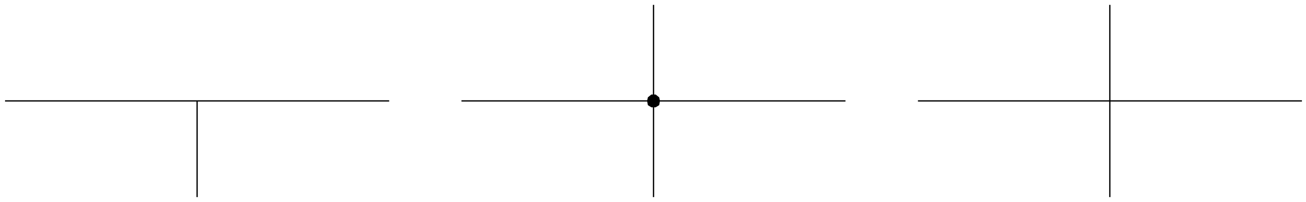
Circuit Schematic Rules (cont.)

- Wires always connect at a T junction
- A dot where wires cross indicates a connection between the wires
- Wires crossing *without* a dot make no connection

wires connect at a T junction

wires connect at a dot

wires crossing without a dot do not connect

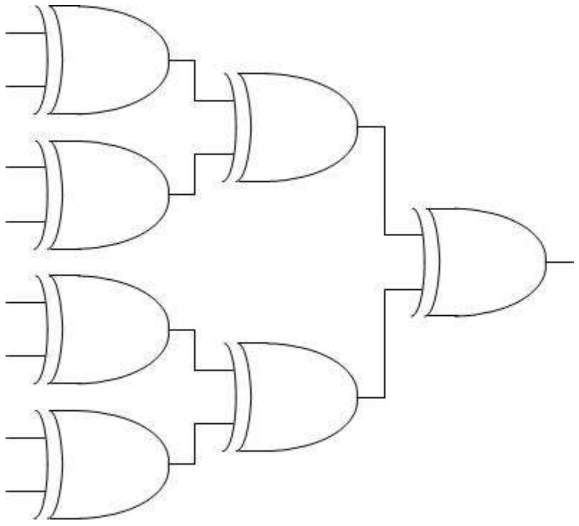


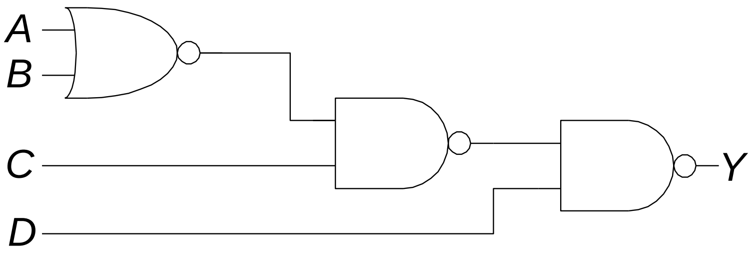
Two-Level Logic

- Two-level logic: **ANDs** followed by **ORs**
- Example: $Y = \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$

Multilevel Logic

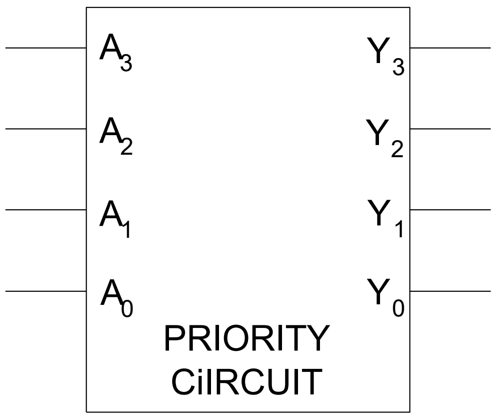
- Complex logic is often built from many stages of simpler gates.





- **Example: Priority Circuit**

Output asserted corresponding to most significant TRUE input



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

Priority Circuit Hardware

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	1	**0**	**0**	**0**	**1**
0	0	1	0	**0**	**0**	**1**	**0**
0	0	1	1	**0**	**0**	**1**	**0**
0	1	0	0	**0**	**1**	**0**	**0**
0	1	0	1	**0**	**1**	**0**	**0**
0	1	1	0	**0**	**1**	**0**	**0**
0	1	1	1	**0**	**1**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**0**
1	0	0	1	**1**	**0**	**0**	**0**
1	0	1	0	**1**	**0**	**0**	**0**
1	0	1	1	**1**	**0**	**0**	**0**
1	1	0	0	**1**	**0**	**0**	**0**
1	1	0	1	**1**	**0**	**0**	**0**
1	1	1	0	**1**	**0**	**0**	**0**
1	1	1	1	**1**	**0**	**0**	**0**

Don't Cares

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	1	**0**	**0**	**0**	**1**
0	0	1	0	**0**	**0**	**1**	**0**
0	0	1	1	**0**	**0**	**1**	**0**
0	1	0	0	**0**	**1**	**0**	**0**
0	1	0	1	**0**	**1**	**0**	**0**
0	1	1	0	**0**	**1**	**0**	**0**
0	1	1	1	**0**	**1**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**0**
1	0	0	1	**1**	**0**	**0**	**0**
1	0	1	0	**1**	**0**	**0**	**0**
1	0	1	1	**1**	**0**	**0**	**0**
1	1	0	0	**1**	**0**	**0**	**0**
1	1	0	1	**1**	**0**	**0**	**0**
1	1	1	0	**1**	**0**	**0**	**0**
1	1	1	1	**1**	**0**	**0**	**0**

$$Y_3 = A_3$$
$$Y_2 = \overline{A_3}\,A_2$$
$$Y_1 = \overline{A_3}\,\overline{A_2}\,A_1$$
$$Y_0 = \overline{A_3}\,\overline{A_2}\,\overline{A_1}\,A_0$$

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0

Two-Level

Logic Forms

Two-Level Logic Variations

- **ANDs** followed by **ORs**: **SOP** form
- **ORs** followed by **ANDs**: **POS** form
- Only **NAND** gates: **SOP** form
- Only **NOR** gates: **POS** form

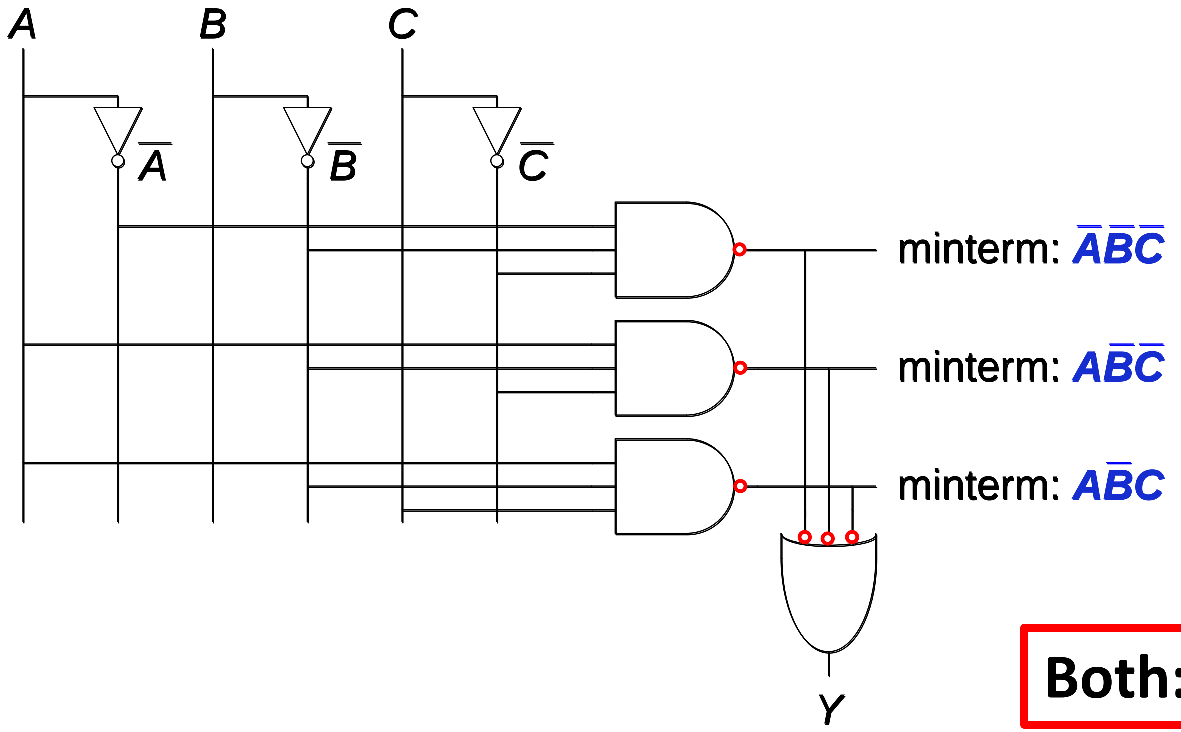
Most common form of two-level logic

Two-Level Logic Variation

- Two-level logic variation: **ORs** followed by **ANDs**
- Example: $Y = (\overline{A}+\overline{B})(A+B+\overline{C})$

Two-Level Logic

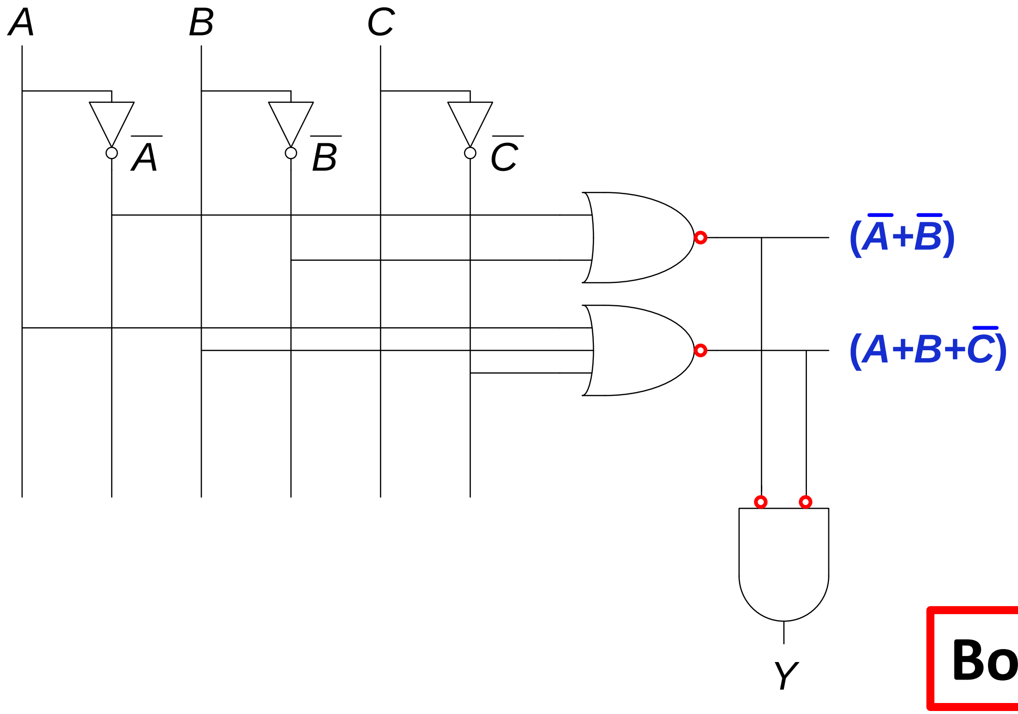
- Two-level logic: **ANDs** followed by **ORs** → **NANDs**
- Example: $Y = \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$



Both: SOP form

Two-Level Logic Variation

- Two-level logic: **ORs** followed by **ANDs** → **NORs**
- Example: $Y = (\overline{A}+\overline{B})(A+B+\overline{C})$



Both: POS form

Bubble Pushing

De Morgan's Theorem

#	Theorem	Dual	Name
T12	$\overline{B \bullet C \bullet D \ldots} = \overline{B} + \overline{C} + \overline{D} \ldots$	$\overline{B + C + D \ldots} = \overline{B} \bullet \overline{C} \bullet \overline{D} \ldots$	De Morgan's Theorem

De Morgan's Theorem

Example D1:

$$Y = \overline{A + \overline{BC}}$$

$$= \overline{A} \bullet \overline{\overline{BC}}$$

$$= \overline{A} \bullet BC$$

$$= \overline{A}BC$$

- Work from the **outside in** (i.e., top bar, then down)
- Use **involution** when possible

DeMorgan’s Theorem

Example D2:

$$Y = \overline{A + \overline{BC} + \overline{A}\,\overline{B}}$$

$$= \overline{A} \bullet \overline{\overline{BC}} \bullet \overline{\overline{A}\,\overline{B}}$$

$$= \overline{A} \bullet BC \bullet (\overline{\overline{A}} + \overline{\overline{B}})$$

$$= \overline{A}BC \bullet (A + B)$$

$$= \overline{A}BCA + \overline{A}BCB$$

$$= \overline{A}BC$$

- De Morgan **applies to**:
  - **Products** under a bar
  - **Sums** under a bar
- **Do not** try to apply DeMorgan’s to a **mix of operations**

De Morgan's Theorem

Example D2:

$$Y = \overline{A + \overline{BC} + \overline{A}\,\overline{B}}$$

$$= \overline{A} \bullet \overline{\overline{BC}} \bullet \overline{\overline{A}\,\overline{B}}$$

$$= \overline{A} \bullet BC \bullet (\overline{\overline{A}} + \overline{\overline{B}})$$

$$= \overline{A}BC \bullet (A + B)$$

$$= \overline{A}BCA + \overline{A}BCB$$

$$= \overline{A}BC$$

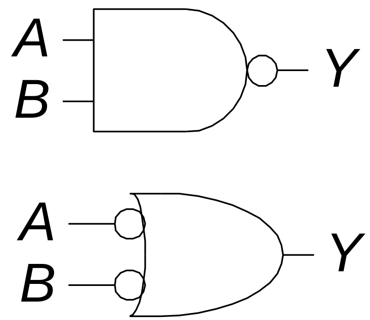
Don't forget these parentheses!

Remember:

$$\mathbf{\overline{AB} = (\overline{A} + \overline{B})}$$

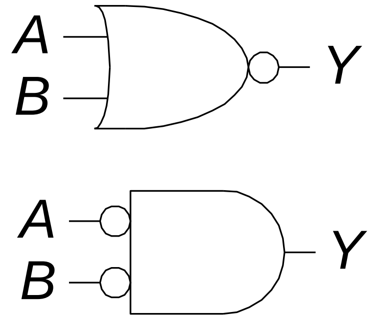
De Morgan's Theorem: Gates

- $Y = \overline{AB} = \overline{A} + \overline{B}$



NAND gate
two forms

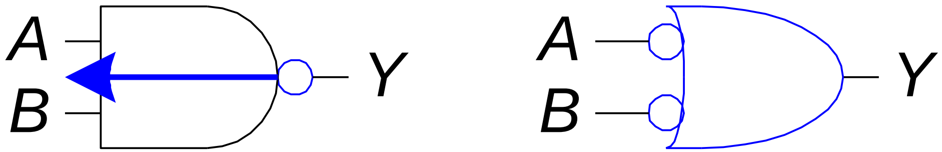
- $Y = \overline{A + B} = \overline{A} \cdot \overline{B}$



NOR gate
two forms

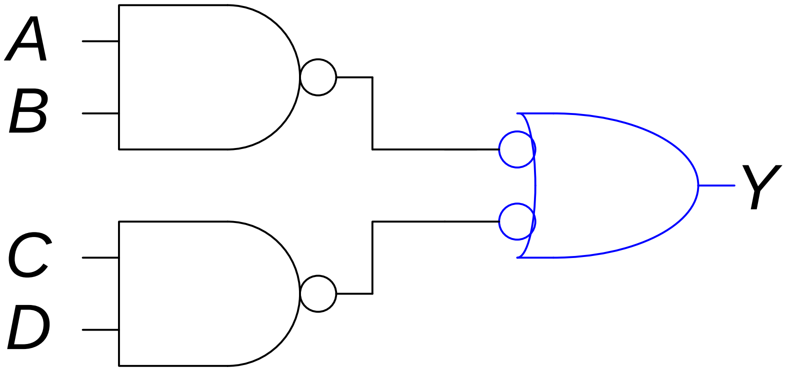
Bubble Pushing

- **Backward:**
  - Body changes
  - Adds bubbles to inputs



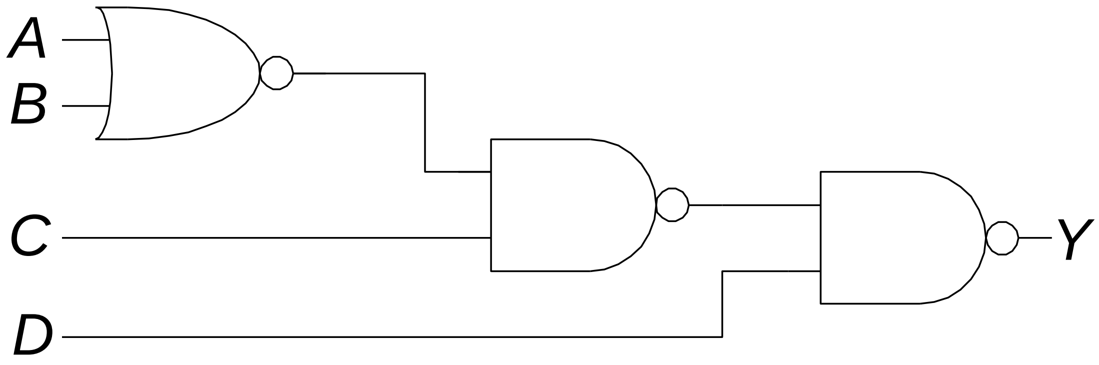
Bubble Pushing

- What is the Boolean expression for this circuit?

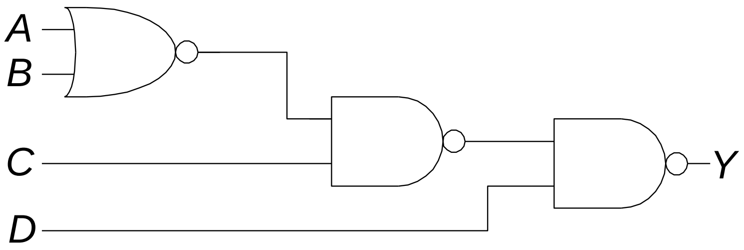


Bubble Pushing Rules

- Begin at output, then work toward inputs
- Push bubbles on final output back
- Draw gates in a form so bubbles cancel



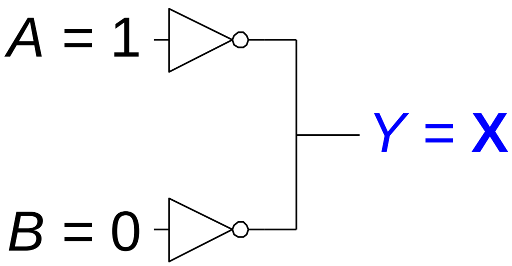
Bubble Pushing Example



X’s and Z’s, Oh My

Contention: X

- **Contention:** circuit tries to drive output to 1 **and** 0
  - Actual value somewhere in between
  - Could be 0, 1, or in forbidden zone
  - Might change with voltage, temperature, time, noise
  - Often causes excessive power dissipation

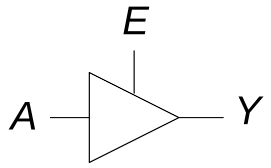


- **X is also used for:**
  - **Uninitialized values**
  - **Don't Care**
- **Warnings:**
  - Contention or uninitialized outputs usually indicate a **bug**.
  - **L**ook at the context to tell meaning

Floating: Z

- Floating, high impedance, open, high Z
- Floating output might be 0, 1, or somewhere in between
  - A voltmeter **won't** indicate whether a node is floating
  - But if you touch the node or your instructor walks over for a checkoff, it may change randomly

Tristate Buffer

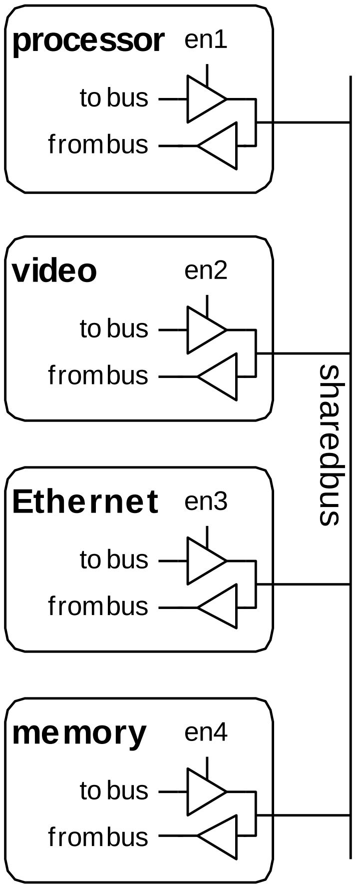


E	*A*	*Y*
0	0	Z
0	1	Z
1	0	0
1	1	1

Tristate Busses

Floating nodes are used in tristate busses

- Many different drivers
- Exactly one is active at once



Karnaugh Maps

Karnaugh Maps (K-Maps)

- Boolean expressions can be minimized by combining terms
- K-maps minimize equations graphically
  - $PA + P\overline{A} = P$

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Y (C \ AB)	00	01	11	10
0				
1				

Y (C \ AB)	00	01	11	10
0	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\,\overline{C}$
1	$\overline{A}\,\overline{B}C$	$\overline{A}BC$	ABC	$A\overline{B}C$

K-Map

- Circle 1’s in adjacent squares
- In Boolean expression: include only literals whose true **and** complement form are ***not*** in the circle

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Y: C \ AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

Y: C \ AB	00	01	11	10
0	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\,\overline{C}$
1	$\overline{A}\,\overline{B}C$	$\overline{A}BC$	ABC	$A\overline{B}C$

$$Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C = \overline{A}\,\overline{B}$$

3-Input K-Map

- Circle 1's in adjacent squares
- In Boolean expression: include only literals whose true **and** complement form are ***not*** in the circle

Truth Table

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

K-Map

Y / $C \backslash AB$	00	01	11	10
0				
1				

$Y =$

Some Definitions

- **Complement**: variable with a bar over it

  $\bar{A}, \bar{B}, \bar{C}$

- **Literal**: variable or its complement

  $A, \bar{A}, B, \bar{B}, C, \bar{C}$

- **Implicant**: product of literals

  $AB\bar{C}, \bar{A}C, BC$

- **Prime implicant:** implicant corresponding to the **largest circle** in a K-map

K-Map Rules

- **Every 1 must be circled** at least once
- Each circle must span a **power of 2** (i.e. 1, 2, 4) squares in each direction
- Each circle must be as **large** as possible
- A circle may **wrap around the edges**

4-Input K-Map

A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y

CD \ *AB*	00	01	11	10
00				
01				
11				
10				

4-Input K-Map

A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y

CD \ *AB*	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

Y =

Karnaugh Maps with Don’t Cares

K-Map Rules

- **Every 1 must be circled** at least once
- Each circle must span a **power of 2** (i.e. 1, 2, 4) squares in each direction
- Each circle must be as **large** as possible
- A circle may **wrap around the edges**
- Circle a **“don't care”** (X) **only if it helps** minimize the equation

K-Maps with Don't Cares

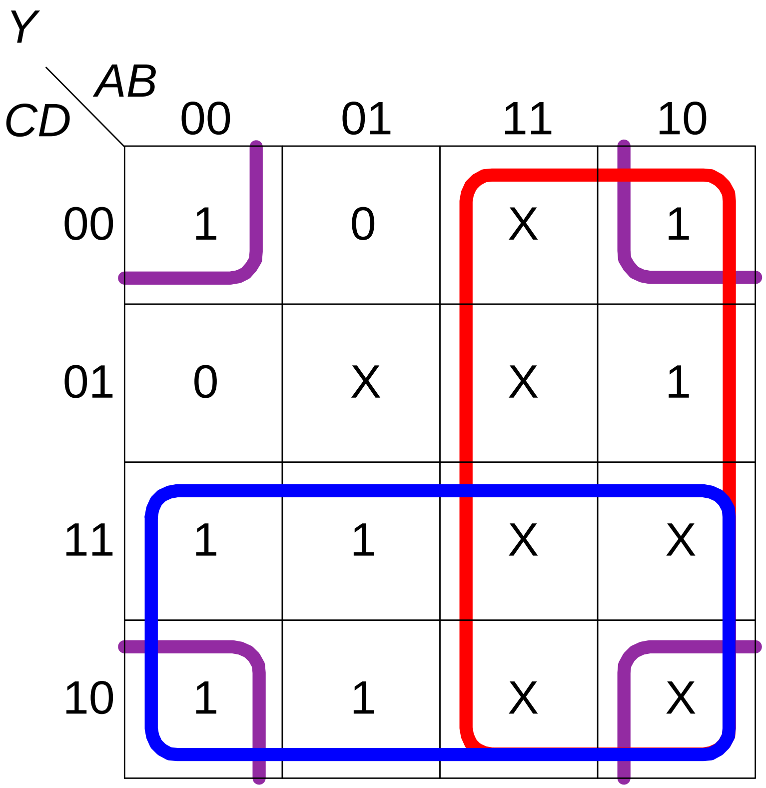
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y

CD \ AB	00	01	11	10
00				
01				
11				
10				

K-Maps with Don't Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

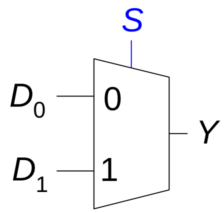


$Y =$

Combinational Building Blocks: Multiplexers

Multiplexer (Mux)

- Selects between one of *N* inputs to connect to output
- **Select** input is $\mathbf{log_2}\boldsymbol{N}$ **bits** – control input
- **Example:** 2:1 Mux



S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

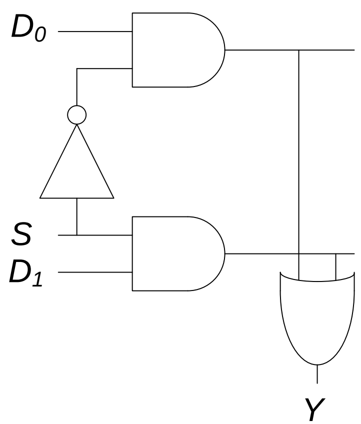
S	Y
0	D_0
1	D_1

2:1 Multiplexer Implementations

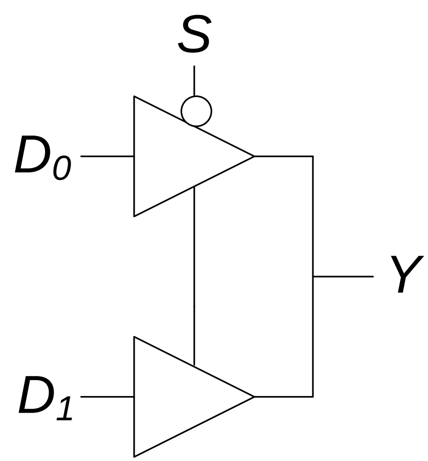
- **Logic gates**
  - Sum-of-products form

S \ D_0D_1	00	01	11	10
0	0	0	1	1
1	0	1	1	0

$$Y = D_0\overline{S} + D_1S$$

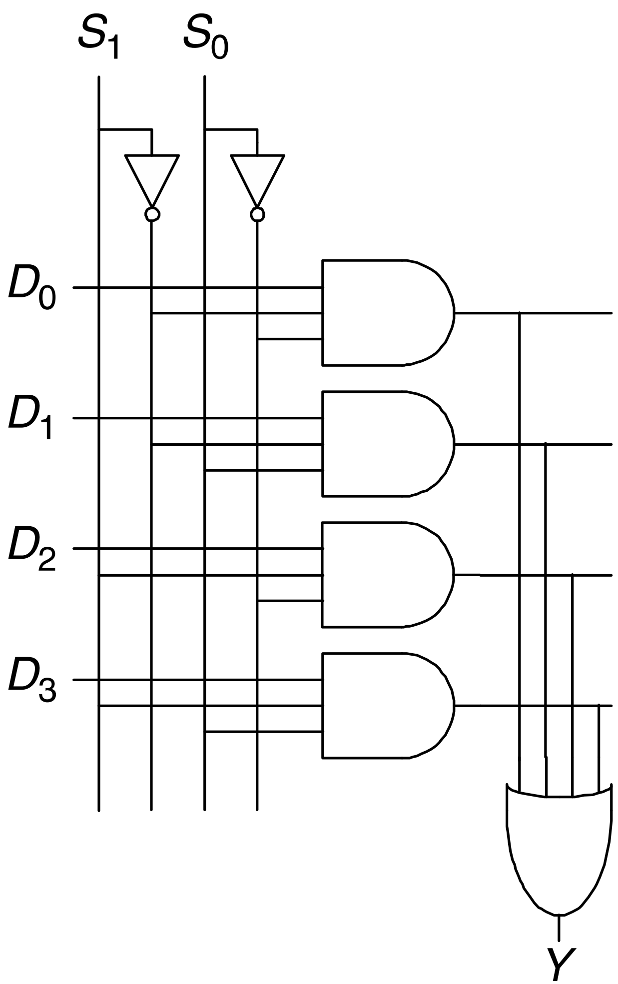


- **Tristates**
  - Two tristates
  - Turn on exactly one to select the appropriate input

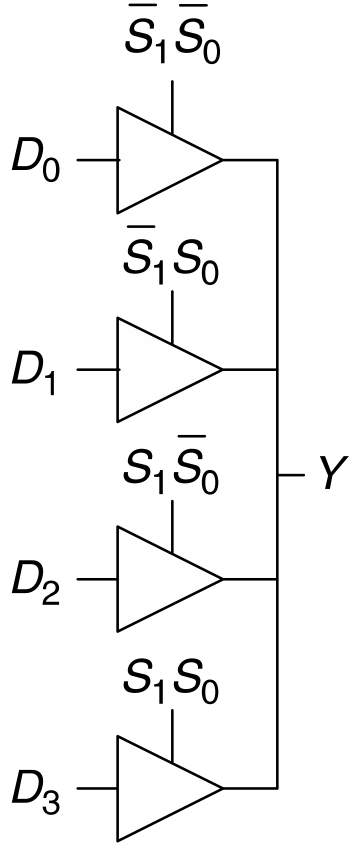


4:1 Multiplexer Implementations

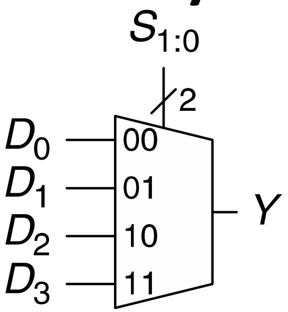
2-Level Logic



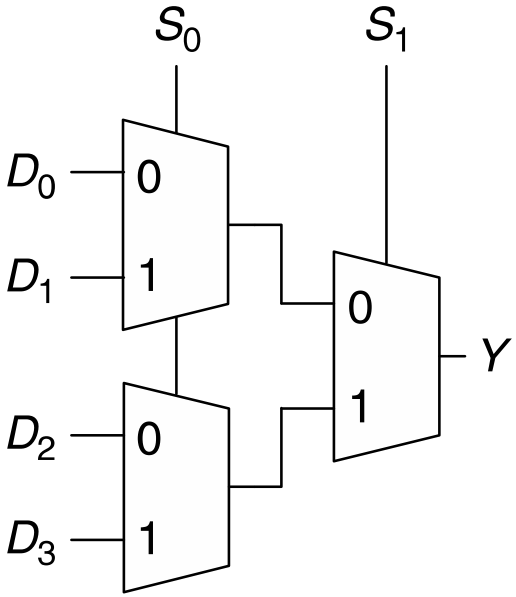
Tristates



4:1 Mux Symbol



Hierarchical

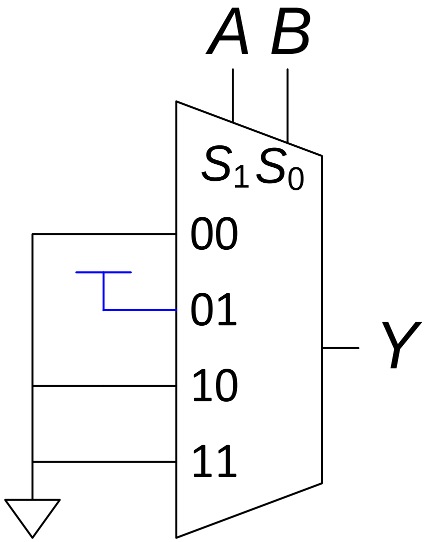


Logic using Multiplexers

Using mux as a **lookup table**

A	B	Y
0	0	0
0	1	1
1	0	0
1	1	0

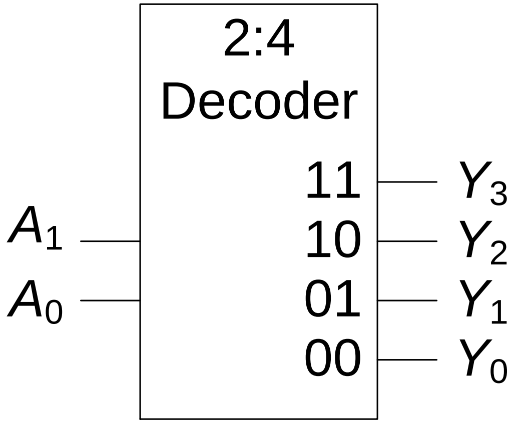
$$Y = \overline{A}B$$



Combinational Building Blocks: Decoders

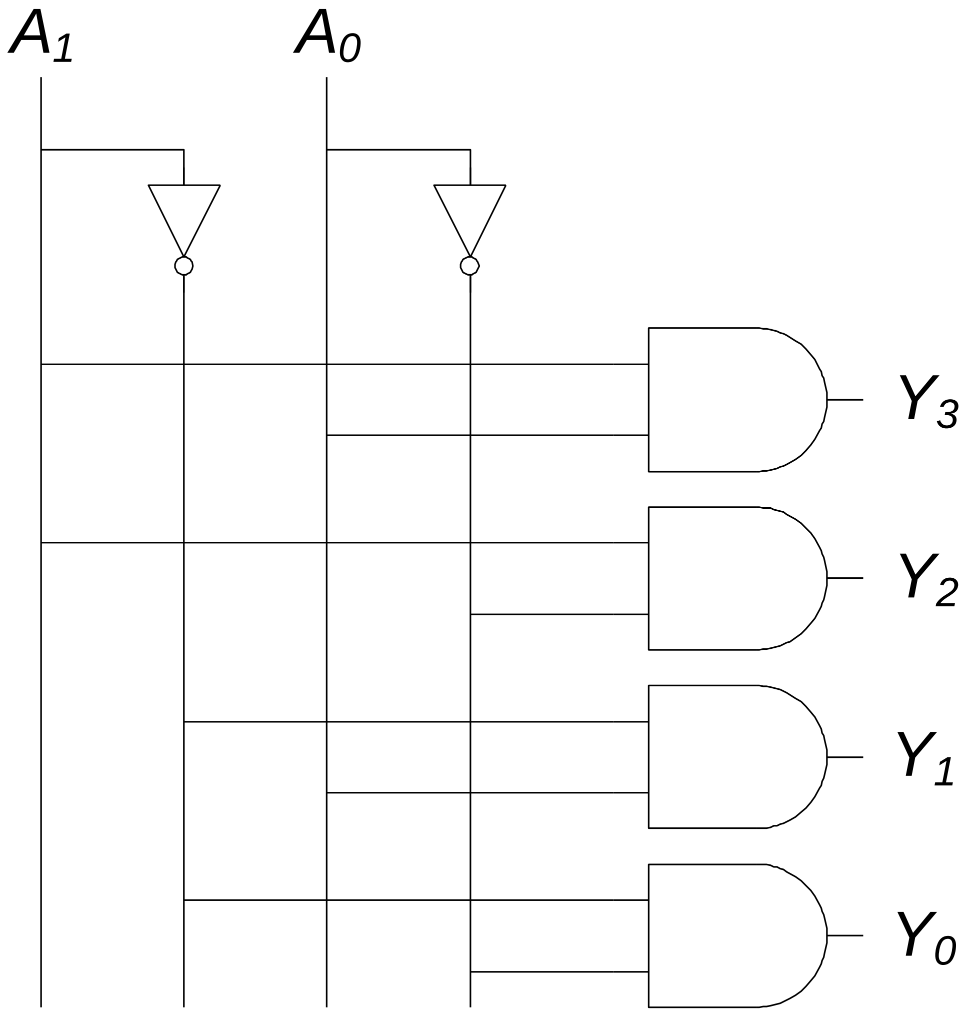
Decoders

- N inputs, 2^N outputs
- **One-hot outputs:** only one output HIGH at once



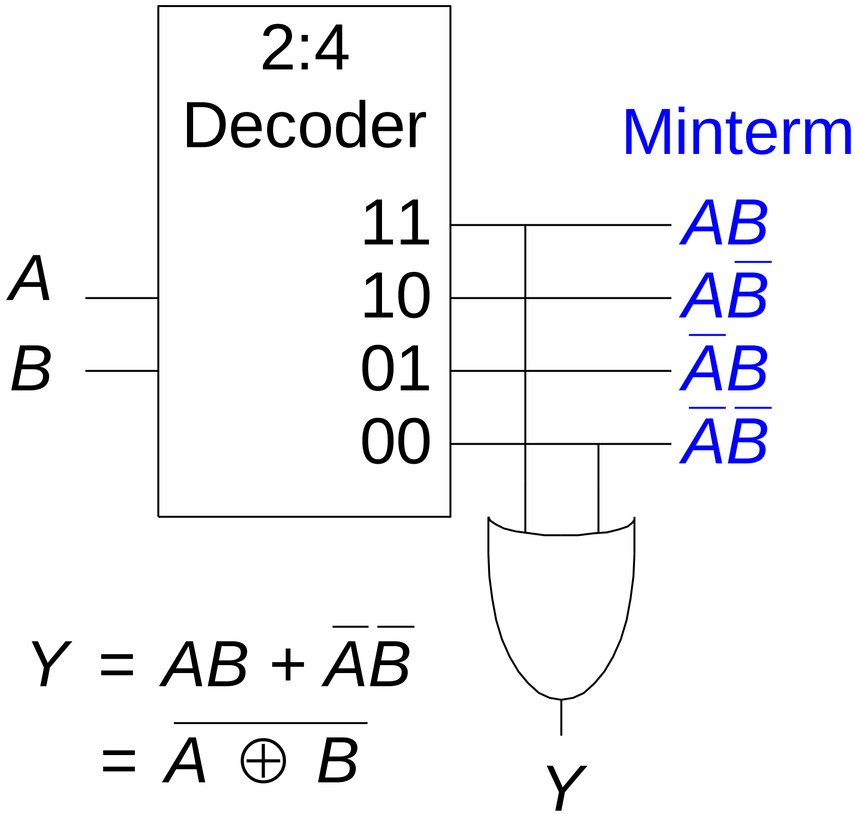
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	**1**
0	1	0	0	**1**	0
1	0	0	**1**	0	0
1	1	**1**	0	0	0

Decoder Implementation



Logic Using Decoders

OR the minterms:

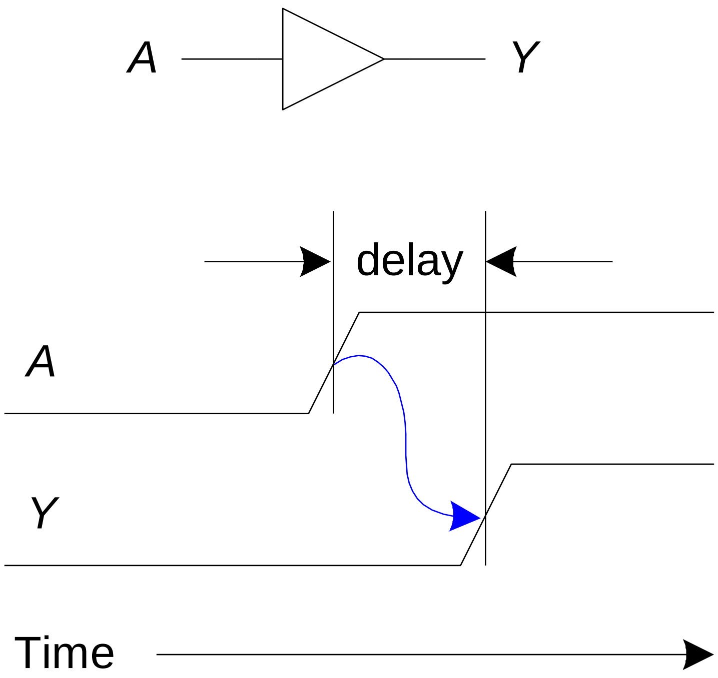


$$Y = AB + \overline{A}\,\overline{B}$$
$$= \overline{A \oplus B}$$

Timing

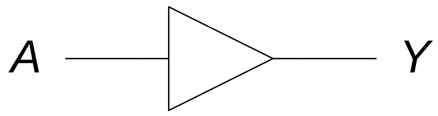
Timing

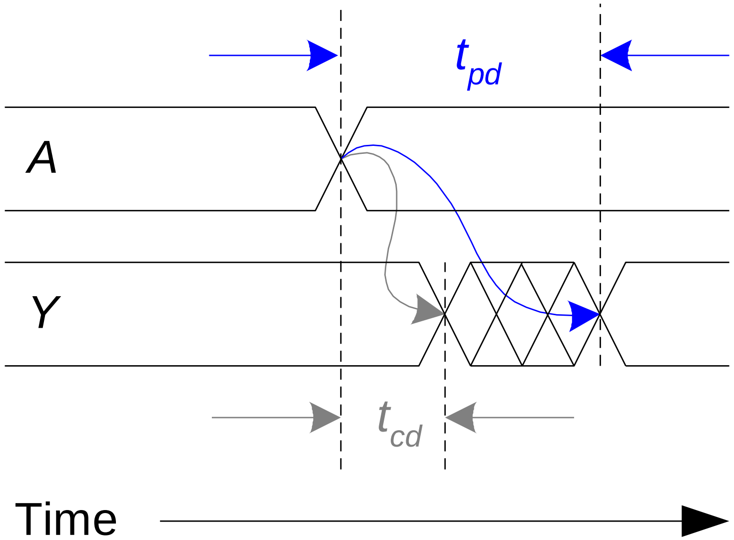
- **Delay:** time between input change and output changing
- How to build fast circuits?



Propagation & Contamination Delay

- **Propagation delay:** t_{pd} = **max** delay from input to output
- **Contamination delay:** t_{cd} = **min** delay from input to output

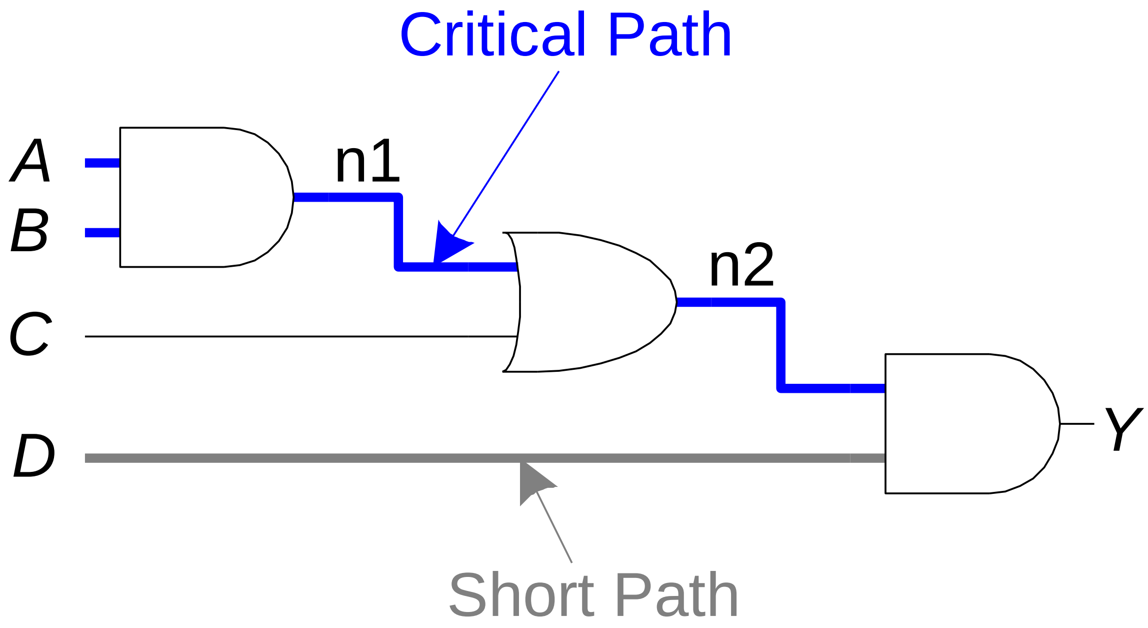




Propagation & Contamination Delay

- **Delay is caused by**
  - Capacitance and resistance in a circuit
  - Speed of light limitation
- **Reasons why t_{pd} and t_{cd} may be different:**
  - Different rising and falling delays
  - Multiple inputs and outputs, some of which are faster than others
  - Circuits slow down when hot and speed up when cold

Critical (Long) & Short Paths



Critical (Long) Path: $t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$ **(max delay)**

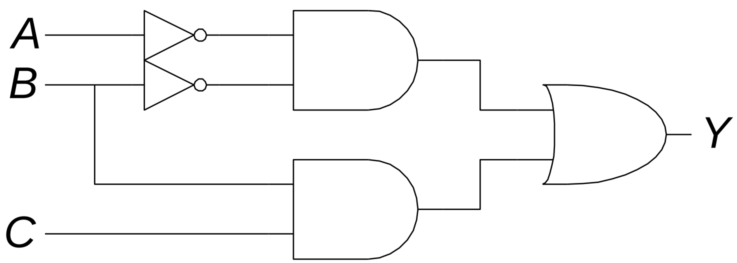
Short Path: $t_{cd} = t_{cd\_AND}$ **(min delay)**

Glitches

When a single input change causes an output to change multiple times

Glitch Example

What happens when A = 0, C = 1, B falls?

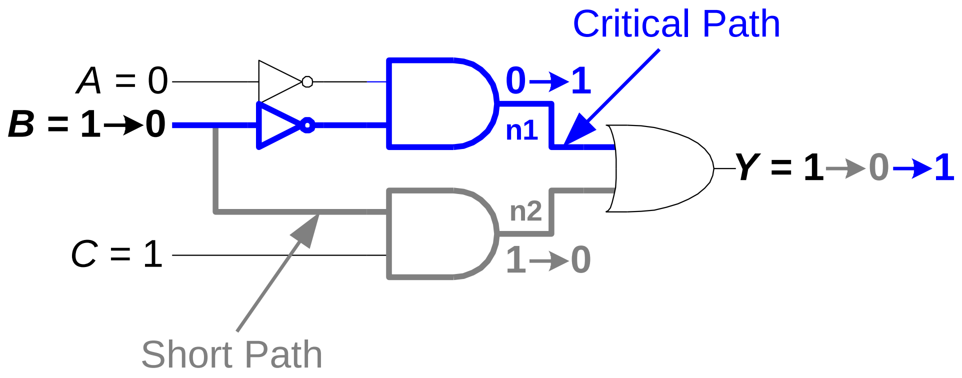


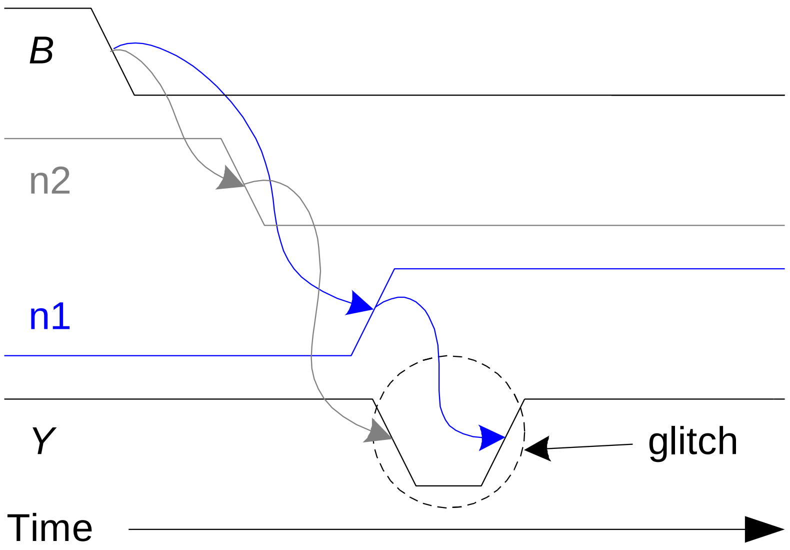
Y

C \ AB	00	01	11	10
0	1	0	0	0
1	1	1	1	0

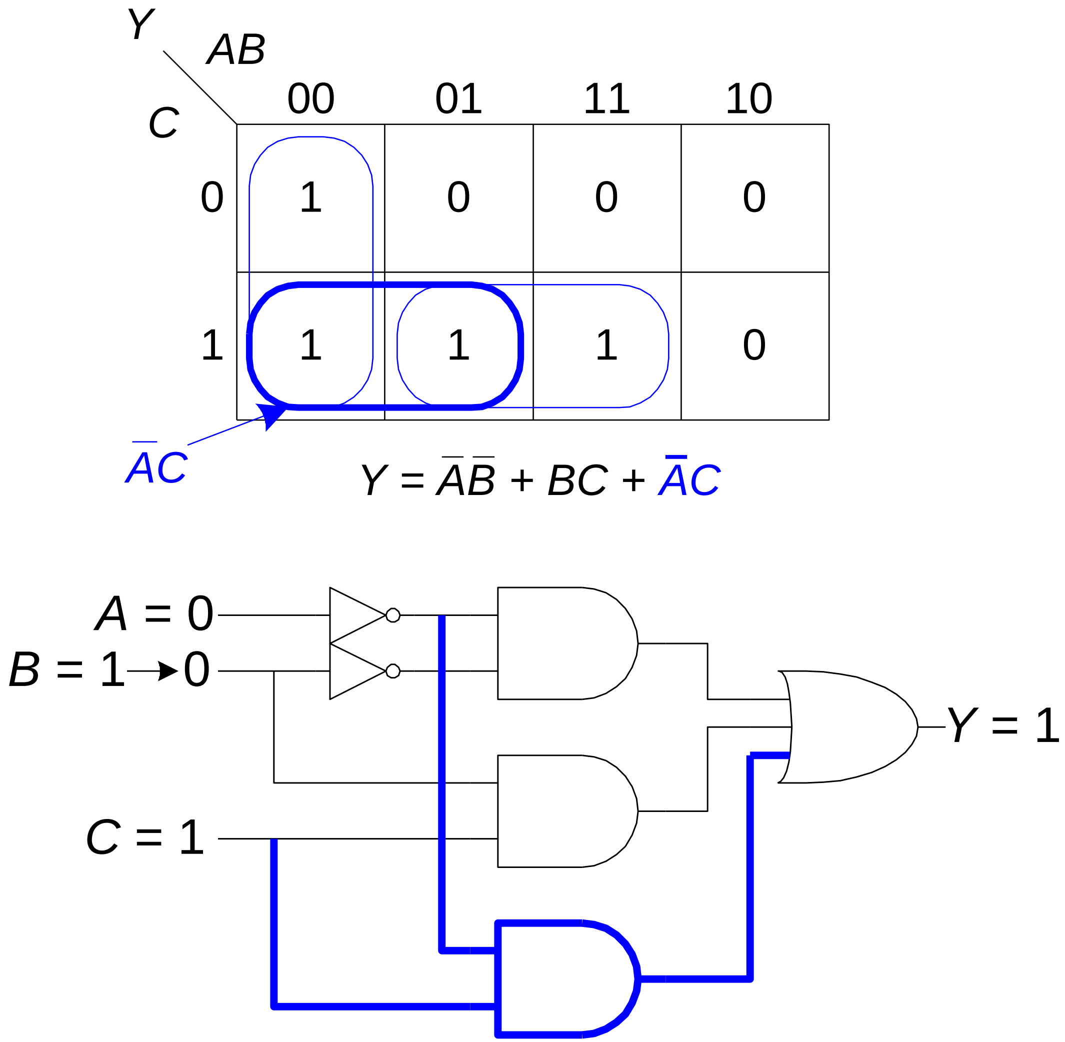
$$Y = \bar{A}\bar{B} + BC$$

Glitch Example (cont.)





Fixing the Glitch



Why Understand Glitches?

- Because of **synchronous design** conventions (see Chapter 3), glitches don't cause problems.
- It's important to **recognize** a glitch: in simulations or on oscilloscope.
- We **can't get rid of all glitches** – simultaneous transitions on multiple inputs can also cause glitches.

About these Notes

Digital Design and Computer Architecture Lecture Notes